ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

Salp Swarm Algorithm Untuk Meminimasi Konsumsi Energi Pada Penjadwalan Flow Shop Dengan Set Up Dan Removal Time

Dana Marsetiya Utama[1]
[1]Universitas Muhammadiyah Malang, Malang

Kontak Person:
Dana Marsetiya Utama
Jalan Raya Tlogomas 246 Malang, 0341-464318
E-mail: dana@umm.ac.id

Abstract

Baru-baru ini, sektor industri menghasilkan sekitar setengah dari total konsumsi energi dunia. Perusahaan manufaktur diharuskan mengurangi konsumsi energi. Karena itu, artikel ini bertujuan untuk mengusulkan Salp Swarm Algorithm (SSA) untuk meminimalkan konsumsi energi pada kasus permutation flow shop scheduling problem(PFSSP). Penelitian ini juga mempertimbangkan waktu setup dan removal. Selanjutnya, SSA dibandingkan dengan beberapa algoritma. Kami menggunakan eksperimen numerik untuk menunjukkan kinerja algoritma yang diusulkan. Analisis komparatif dengan beberapa algoritma sebelumnya telah dilakukan dengan berbagai variasi masalah PFSSP. Berdasarkan percobaan numerik, SSA terbukti kompetitif dibandingkan dengan algoritma lainnya.

Kata kunci: *Flow Shop, Konsumsi Energi, Swarm Salp Algorithm*

1. Pendahuluan

Dalam beberapa dekade terakhir, permintaan energi di dunia telah meningkat secara signifikan. Penggunaan sumber daya energi berbahan bakar fosil masih mendominasi pasokan energi perusahaan [1]. Pada akhir tahun 2040, total permintaan energi di seluruh dunia diperkirakan meningkat sebesar 37% [2]. Penggunaan energi yang berlebihan menimbulkan gas rumah kaca [3]. Aktivitas ini memiliki konsekuensi seperti perubahan iklim. Karena itu, masalah penghematan energi telah menarik lebih banyak perhatian. Saat ini, Sektor perindustrian mengonsumsi setengah dari total konsumsi energi dunia [4]. Dengan demikian, perusahaan manufaktur menjadi sumber utama pemanasan global. Perusahaan manufaktur dituntut mencari cara untuk mengurangi konsumsi energi dan karbon emisi yang dihasilkan [5] [6]. Salah satu strategi untuk mengurangi konsumsi energi dan karbon emisi adalah dengan pejadwalan yang tepat. Penjadwalan dapat memainkan peran penting dalam mengatasi masalah konsumsi energi dan karbon emisi pada sektor manufaktur [7]. Penjadwalan merupakan pengalokasian sumber daya untuk menyelesaikan pekerjaan untuk dikelola secara efisien [8] [9]. Salah satu masalah di penjadwalan adalah permutation flow shop scheduling problem (PFSSP). Beberapa ahli mengklaim kasus PFSSP tidak dapat diselesaikan dalam waktu polinomial. Dengan demikian, PFSSP termasuk dalam masalah NP-Hard [10].

Umumnya, masalah penjadwalan produksi digunanak untuk meminimasi masalah makespan [11] [12]dan tardiness [13], [14]. Saat ini, banyak peneliti menggunakan penjadwalan untuk meminimasi konsumsi energi. Konsumsi energi pada proses produksi terjadi saat proses produksi berlangsung. Namun, sebagian besar energi juga dikonsumsi pada saat mesin dalam keadaan menganggur atau idle [15]. Salah satu strategi penjadwalan untuk mengatasi permasalahan ini adalah strategi ON – OFF [16]. Namun, strategi ini tidak efisien, karena strategi ini menyebabkan umur mesin menjadi pendek. Beberapa penelitian terdahulu menggunakan prosedur heuristik untuk menyeleseikan masalah ini seperti NEH dan CDS [17] dan neighborhood search [2]. Selain itu, Beberapa penelitian menggunakan Hybrid Metaheuristics [18], Genetic Algorithm [19], Hybrid Genetic Algorithm [20], hybrid multi-objective backtracking search algorithm [21], Particle Swarm Optimization (PSO) [22], Adaptive genetic algorithm [23], collaborative optimization algorithm (COA) [24], Harmony-inspired genetic algorithm [25], memetic differential evolution [1]. Saat ini, algoritma metaheuristik digunakan sebagai teknik utama untuk mendapatkan solusi optimal dari masalah optimasi [26].

Sejauh ini, tidak ada penelitian sebelumnya yang meneliti minimisasi konsumsi energi menggunakan algoritma Salp Swarm Algorithm (SSA). Peneliti mengusulkan algoritma Algoritma Salp Swarm (SSA) sebagai pendekatan alternatif untuk memecahkan minimisasi konsumsi energi pada kasus

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

PFSSP dengan mempertimbangkan setup dan removal time. Algoritma Salp Swarm (SSA) adalah algoritma optimasi Penjadwalan Metaheuristik terbaru dan baru yang diusulkan oleh Mirjalili et al [27]. algoritma SSA terinsprasi dari perilaku salp dialam yang memiliki karakter mirip dengan ubur-ubur yang bergerak menuju sumber makanan. Salp biasanya ditemukan dalam kelompok (Swarms) yang disebut rantai Salp. Setiap rantai salp terdiri dari pemimpin dan pengikut. Algoritma SSA digunakan untuk meminimasi konsumsi energi yang digunakan selama proses produksi. Oleh karena itu, tujuan dari penelitian adalah untuk mengembangkan algoritma SSA untuk meminimalkan konsumsi energi di PFSSP dengan mempertimbangkan setup dan removal time.

2. Metode Penelitian

2.1. Asumsi masalah

Permasalahan penjadwalan PFSSP memiliki asumsi,(1) sejumlah *n job* (*n* = 1,2,3...i) dengan urutan yang sama dikerjakan pada serangkaian *m* mesin (*m* = 1,2,3...j); (2) Waktu proses tij merupakan waktu penyelesaian *job* ke-*i* pada mesin ke-*j*; (3) semua mesin tersedia pada t=0; (4) waktu set-up independent terhadap urutan pekerjaan; (5) Sijk adalah Waktu set up untuk berpindah dari job j ke job k pada mesin i; (6) untuk waktu removal terpisah dengan waktu proses; (7) tiap job ketika dimulai akan diproses sampai selesei (tidak bisa disela); (8) tiap mesin berhenti ketika job terakhir pada tiap mesin selesei (tiap mesin yang berhenti independent dari mesin yang lain).

2.2. Definisi masalah

Model PFSSP telah dimodifikasi dari Li, et al. [28]. Penjadwalan terbaik didefinisikan memiliki TEC minimum. Model PFSSP untuk meminimalkan konsumsi energi adalah sebagai berikut

Objective function $Z = \min TEC$ (1)

Subject to :

$$C_{1,1} = S_1 + P_{1,1} + R_{1,1} \quad (2)$$

$$C_{1,j} = \max\left(C_{1,j-1} - R_{1,j-1}, S_1 \right) + P_{1,j} + R_{1,j} \ , \quad j = 2..m \quad (3)$$

$$C_{i,1} = C_{i-1,1} + S_{i-1,i} + P_{i,1} + R_{i,1} \ , \qquad i = 2..n \quad (4)$$

$$C_{i,j} = \max\left(C_{i,j-1} - R_{i,j-1}, S_{i-1,i} + C_{i-1,j}\right) + P_{i,j} + R_{i,j} \ , i = 2..n, j = 2..m \quad (5)$$

$$Bj = \sum_{i=1}^{n} P_{i,j} \ , \quad \forall j = 1..m \quad (6)$$

$$Sj = \sum_{i=2}^{n} S_{i-1,i} + S_1 \ , \quad \forall j = 1..m \quad (7)$$

$$Rj = \sum_{i=1}^{n} R_{i,j} \quad \forall j = 1..m \quad (8)$$

$$Tj = \max\left(C_{i,j}\right) \ , \forall i = 1..n, \ j = 1..m \quad (9)$$

$$Ij = Tj - Bj - Sj - Rj \ , \forall j = 1..m \quad (10)$$

$$TEC = \sum_{j=1}^{m} (Bj.Pej + Ij.Iej + Sj.Sej + Rj.Rej) \quad (11)$$

Persamaan (1) menggambarkan permutasi TEC (fungsi tujuan); Pembatas Persamaan (2) menjelaskan waktu penyelesaian urutan kerja satu pada mesin 1; Pembatas Persamaan (3) menjelaskan bahwa mesin 2 sampai m; Pembatas Persamaan (4) menjelaskan waktu penyelesaian urutan saya bekerja dari mesin 1; Pembatas Persamaan (5) menunjukkan bahwa mesin j; Pembatas Persamaan (6) menjelaskan waktu sibuk total mesin; Pembatas Persamaan (7) menjelaskan waktu pengaturan total. Pembatas Persamaan (8) menggambarkan total waktu penghapusan. Pembatas Persamaan (9) menunjukkan waktu penyelesaian mesin j dari permutasi; Pembatas Persamaan (10) menunjukkan total waktu idle dari mesin permutasi j; dan Pembatas Persamaan (11) menjelaskan model PFSSP untuk konsumsi energi.

2.3. Algoritma Salp Swarm (SSA)

Algoritma Salp Swarm (SSA) adalah algoritma optimasi Penjadwalan Metaheuristik terbaru dan baru yang diusulkan oleh Mirjalili et al [27]. SSA terinspirasi dari perilaku dari salps di lautan. SSA telah diterapkan untuk memecahkan sejumlah besar masalah optimisasi matematika dan banyak masalah mengenai desain teknik [29]. Hasil penerapan SSA juga telah membuktikan dalam masalah pemilihan

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

fitur dibandingkan dengan algoritma lain seperti PSO dan DE. SSA juga diterapkan untuk mengatasi pemilihan fitur dan membuktikan kinerja yang baik [27]. SSA merupakan algoritma pertama yang meniru perilaku Salps di alam mereka mirip dengan ubur-ubur di jaringan mereka dan bergerak menuju sumber makanan [30]. Salp biasanya ditemukan dalam kelompok (Swarms) yang disebut rantai Salp; setiap rantai salp berisi pemimpin dan pengikut [31].

Pemimpin adalah salp di depan rantai, sedangkan salp lainnya dianggap sebagai pengikut. Posisi salps didefinisikan dalam ruang pencarian n-dimensi di mana n adalah jumlah variabel dari masalah yang diberikan. Oleh karena itu, posisi semua salps disimpan dalam matriks dua dimensi yang disebut x. Juga diasumsikan bahwa ada sumber makanan yang disebut F di ruang pencarian sebagai target kawanan. Untuk memperbarui posisi pemimpin, persamaan berikut adalah diusulkan:

$$X_j^i = \begin{cases} F_j + C_1 \left((ub_j - lb_j)c_2 + lb_j \right) & c_3 \geq 0 \\ F_j - c_1 \left((ub_j - lb_j)c_2 + lb_j \right) & c_3 \geq 0 \end{cases} \quad (12)$$

Dimana x1j menunjukkan posisi salp pertama (pemimpin) di Dimensi j, Fj adalah posisi sumber makanan di jth dimensi, ubj menunjukkan batas atas dimensi j, lbj menunjukkan batas bawah dimensi j, c1, c2, dan c3 adalah angka acak. Eq. (3.1) menunjukkan bahwa pemimpin hanya memperbarui posisinya dengan menghormati sumber makanan. Koefisien c1 adalah parameter terpenting dalam SSA karena menyeimbangkan eksplorasi dan eksploitasi yang didefinisikan sebagai berikut:

$$C_1 = 2e^{-(\frac{4i}{L})^2} \quad (13)$$

Di mana l adalah iterasi saat ini dan L adalah jumlah maksimum iterasi. Parameter c2 dan c3 adalah angka acak yang dihasilkan secara seragam dalam interval [0,1]. Bahkan, mereka menentukan apakah posisi berikutnya dalam dimensi j harus menuju infinity positif atau infinity negatif serta ukuran langkah.

Untuk memperbarui posisi pengikut, persamaan berikut digunakan (hukum gerak Newton):

$$X_j^i = \frac{1}{2} a t^2 + v_0 t \quad (14)$$

Jika i ≥ 2, xij menunjukkan posisi salp pengikutnya di dimensi j, t adalah waktu, v0 adalah kecepatan awal, dan $a = \frac{v_{final}}{v_0}$ dimana $v = \frac{x - x_0}{t}$

Karena waktu dalam optimasi adalah iterasi, perbedaan antara iterasi sama dengan 1, dan mempertimbangkan $V_0 = 0$, persamaan ini dapat dinyatakan sebagai berikut:

$$X_j^i = \frac{1}{2}(X_j^i + X_j^{i-1} \quad (15)$$

Di mana i ≥ 2 dan xij menunjukkan posisi pengikut salp di jth dimensi. Dengan Persamaan. (12) dan (15), rantai salp dapat disimulasikan. Psuodocode Algoritma Salp Swarm (SSA) dapat dilihat pada gambar 1.

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

```
Inisialisasi populasi salp xi (i = 1, 2, ..., n) dengan mempertimbangkan ub dan lb
while (kondisi akhir tidak terpenuhi)
Hitung kebugaran setiap agen pencarian (salp)
F= agen pencarian terbaik
Perbarui c1 oleh Persamaan. (13
   for setiap salp (xi)
      if (i==1)
         Perbarui posisi salp terkemuka dengan Persamaan. (12)
      else
         Perbarui posisi salp pengikut dengan Persamaan. (15)
      end
                                   end
                  Ubah salp berdasarkan batas atas dan bawah variabel
                                   end
                                  kembali F
```

Gambar 1. Psuodocode Algoritma Salp Swarm (SSA)

2.4. Prosedur percobaan

Waktu pemrosesan dihasilkan dari distribusi acak yang seragam (10,50). Waktu setup untuk pekerjaan dalam urutan pertama dihasilkan dari distribusi acak seragam (1,10). Pengaturan waktu untuk pindah dari pekerjaan i-1 ke pekerjaan saya dihasilkan dari distribusi yang seragam (1,10). Waktu penghapusan dihasilkan dari distribusi acak yang seragam (1,5). Konsumsi energi yang dibutuhkan selama operasi pemrosesan dihasilkan dari distribusi acak yang seragam (5,10). Konsumsi energi dihasilkan dari distribusi acak seragam (1,3). Penghapusan konsumsi energi dihasilkan dari distribusi acak seragam (1,3). Mesin idle konsumsi energi dihasilkan dari angka acak seragam (1,2).

Pada artikel ini, percobaan menggunakan aplikasi MATLAB untuk penyelesaian running dengan berbagai kemungkinan. Penelitian mencoba dengan berbagai jumlah populasi, job mesin sampai dengan jumlah iterasi. Pada proses percobaan terdapat 7 variasi jumlah job yaitu 5, 10, 20, 40, 60, 80, dan 100. Untuk variasi atribut mesin terdapat 2 jenis yaitu 4 dan 16 mesin. Lalu untuk atribut jumlah iterasi yang digunakan terdapat 5 jenis, yaitu 10, 50, 100, 200, dan 500 kali. Dan yang terakhir untuk jumlah populasi terdapat 2 jenis yatu 10 dan 100 populasi. Berbagai variasi populasi, job, mesin dan jumlah iterasi digunakan untuk mencari kemungkinan paling optimal untuk meminimasi konsumsi energi. Setelah menngunakan aplikasi MATLAB untuk mencari solusi dengan berbagai paramater, yaitu jumlah populasi dan jumlah iterasi. Jumlah populasi yang digunakan adalah 10, 50, dan 100. Jumlah iterasi yang digunakan adalah 10, 50, 100, 200, dan 500. Selain itu menggunakan beberapa variasi job dan mesin. Selanjutnya, parameter terbaik dari hasil eksperimen dibandingkan dengan beberapa algoritma sebelumnya termasuk Genetic Algorithm (GA) [19] dan particle swarm optimization (PSO) [22]. Kinerja algoritma diukur oleh Efficiency Index Percentage (EIP). EIP digambarkan sebagai persentase rasio konsumsi energi antara algoritma SSA dan algoritma lainnya (persamaan 16).

$$\mathrm{EIP} = \frac{\overline{T}_{\text{proposed algorithm}}}{\overline{T}_{\text{another algorithm}}} \mathrm{x}100\% \qquad (16)$$

3. Hasil dan Pembahasan

Hasil percobaan parameter SSA ditunjukkan pada tabel 1. Ini menunjukkan bahwa semakin tinggi jumlah iterasi dan jumlah populasi yang digunakan, HWOA menghasilkan konsumsi energi yang lebih rendah. Untuk kasus pekerjaan kecil, parameter terbaik adalah menggunakan populasi kecil dan iterasi. Sebaliknya, untuk kasus pekerjaan besar, populasi dan iterasi yang digunakan adalah besar.

Penilaian Efisiensi Indeks Persentase (EIP) konsumsi energi pada tabel 2 membuktikan bahwa SSA memberikan kinerja yang lebih baik dalam kasus-kasus job sedang dan besar. Secara keseluruhan,

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

EIP dari konsumsi energi SSA dibandingkan dengan Genetic Algorithm (GA) [19] dan particle swarm optimization (PSO) [22] memiliki nilai berturut-turut 98.78% dan 98.75%.

Tabel 1. Percobaan parameter SSA terhadap Konsumsi Energi

Job	Mesin	Populasi 10					Populasi 100				
		Jumlah iterasi					Jumlah iterasi				
		10	50	100	200	500	10	50	100	200	500
5	4	6520	6520	6520	6520	6520	6520	6520	6520	6520	6520
5	16	26144	26144	26144	26144	26144	26144	26144	26144	26144	26144
10	4	9438	9438	9438	9438	9438	9438	9438	9438	9438	9438
10	16	49205	49124	48903	48935	48838	48854	48930	48887	48739	48736
20	4	18852	18809	18832	18814	18802	18788	18812	18809	18771	18770
20	16	87022	86518	86769	86705	86566	86909	86498	86561	86418	86377
40	4	39118	39036	39030	39018	39002	39102	39042	38988	39002	39000
40	16	165277	164802	164989	165085	164888	164933	164956	164803	164479	164588
60	4	56470	56341	56374	56341	56399	56425	56362	56387	56374	56344
60	16	232383	232499	232121	232326	231900	232116	231753	232001	231959	231598
80	4	85228	85200	85199	85191	85175	85220	85171	85183	85145	85167
80	16	338282	337943	338013	337516	337703	338081	337733	337387	337435	337172
100	4	102991	102927	102853	102922	102923	102896	102888	102883	102858	102861
100	16	402194	401307	400646	400854	400563	400712	400486	400661	400585	400001

Tabel 2. Perbandingan konsumsi energi dan EIP dari beberapa algoritma lainnya

Job	Mesin	EIP		Job	Mesin	EIP	
		GA	PSO			GA	PSO
5	4	100%	100%	40	16	98.76%	98.44%
5	16	100%	100%	60	4	98.79%	98.53%
10	4	100%	100%	60	16	97.61%	97.29%
10	16	100%	100%	80	4	97.50%	97.15%
20	4	99.58%	99.49%	80	16	97.47%	97.57%
20	16	99.46%	99.63%	100	4	97.52%	97.50%
40	4	99.14%	99.63%	100	16	97.13%	97.27%
Rata-rata						98.78%	98.75%

4. Kesimpulan

Dalam artikel ini, penelitian ini mengusulkan algoritma SSA untuk meminimasi konsumsi energi. penelitian ini mengusulkan parameter terbaik untuk menyelesaikan kasus konsumsi energi. Dalam kasus pekerjaan kecil, lebih baik menggunakan populasi dan iterasi kecil. Sebaliknya, untuk kasus pekerjaan besar, lebih baik menggunakan populasi dan iterasi yang signifikan. Selanjutnya, SSA dibandingkan dengan beberapa prosedur. Eksperimen komputasi membuktikan bahwa SSA menghasilkan konsumsi energi yang optimal. Beberapa area penelitian dapat dipelajari untuk pekerjaan di masa depan. Penelitian ini mengusulkan bahwa SSA dapat digunakan sebagai solusi awal untuk algoritma metaheuristik lainnya. Akhirnya, algoritma SSA yang ditawarkan dapat diterapkan untuk mengurangi konsumsi energi dalam masalah PFSSP.

Daftar Notasi

i : index of jobs, i $= 1,2 \ldots, n$
j : index of machines, j $= 1,2 \ldots, m$
n : total number of jobs
Ci,j : completion time of job sequence i at on machines j
Tj : completion time of machines j

m	:	total number of machines
Pi,j	:	processing time of job sequence i on machines j
Si	:	Setup time of job i in the first sequence on every machine
$S_{i-1,i}$	:	set up time move sequence $i-1$ to i on machine
Ri,j	:	waktu removal untuk job i pada mesin j
Rej	:	energy consumption index of machine j when removal
Pej	:	energy consumption index of machine j
Sej	:	energy consumption Setup index of machine j
Iej	:	energy consumption index of machine j when idle
Bj	:	total busy time of machines j
Ij	:	total idle time of machines j
Sj	:	total setup time of machines j
Rj	:	Removal time of machines j
TEC	:	total energy consumption

Referensi

[1] X. Wu and A. Che. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega 82*, 155-165.

[2] X. Wu and A. Che. (2019). Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega*, 102117.

[3] S. K. D. B. Maulana, D. M. Utama, M. S. Asrofi, I. S. Ningrum, N. Alba, H. A. Ahfa, *et al.* (2019). The Capacitated Sustainable EOQ Model: A Model Considering Emission Tax. *Jurnal Teknik Industri* [Inventory; Lot size; EOQ model; Sustainability; Capital Constraints]. *21(1)*, Article In Press.

[4] J.-Y. Ding, S. Song, and C. Wu. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research 248(3)*, 758-771.

[5] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems 30(4)*, 234-240.

[6] D. M. Utama, T. Baroto, D. Maharani, F. R. Jannah, and R. A. Octaria. (2019). Algoritma ant-lion optimizer untuk meminimasi emisi karbon pada penjadwalan flow shop dependent sequence set-up. *2019* [ant lion optimization; penjadwalan; flow shop; dependent sequence setup; emisi karbon]. *9(1)*, 69-78.

[7] D. M. Utama. (2019). An Effective Hybrid Sine Cosine Algorithm to Minimize Carbon Emission on Flow-shop Scheduling Sequence Dependent Setup. *2019* [carbon emissions; hybrid; Sine Cosine algorithm, flow shop]. *20(1)*, 62-72.

[8] R. Nasution, A. K. Garside, and D. M. Utama. (2017). Penjadwalan Job Shop Dengan Pendekatan Algoritma Artificial Immune System. *2017* [artificial immune system, job shop, schedulling]. *18(1)*, 14.

[9] D. M. Utama. (2017). Analisa Perbandingan Penggunaan Aturan Prioritas Penjadwalan Pada Penjadwalan Non Delay N Job 5 Machine. *Prosiding SENTRA (Seminar Teknologi dan Rekayasa) 2*, 19-23.

[10] M. R. Garey, D. S. Johnson, and R. Sethi. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research 1(2)*, 117-129.

[11] D. M. Utama. (2018). Algoritma LPT-Branch and Bound Pada Penjadwalan Flexible Flowshop untuk Meminimasi Makespan. *PROZIMA (Productivity, Optimization and Manufacturing System Engineering) 2(1)*, 20-26.

[12] D. M. Utama, A. K. Garside, and W. Wicaksono. (2019). Pengembangan algoritma Hybrid Flow shop Three-Stage Dengan Mempertimbangkan Waktu Setup. *Jurnal Ilmiah Teknik Industri 18(1)*, 72-78.

[13] A. K. Garside, D. M. Utama, and M. R. Arifin. (2018). Penjadwalan produksi flowshop menggunakan algoritma branch and bound untuk meminimasi mean tardiness. *2018 3*, 1-6.

[14] D. M. Utama, L. R. Ardiansyah, and A. K. Garside. (2019). Penjadwalan Flow Shop untuk Meminimasi Total Tardiness Menggunakan Algoritma Cross Entropy–Algoritma Genetika. *Jurnal Optimasi Sistem Industri 18(2)*, 133-141.

[15] G. Mouzon, M. B. Yildirim, and J. Twomey. (2007). Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research 45(18-19)*, 4247-4271.

[16] F. R. Tampubolon. (2018). Penggunaan Algoritma Genetika pada Persoalan Multiobjective Flexible Job Shop Scheduling.

[17] D. M. Utama. (2018). Pengembangan Algoritma NEH Dan CDS Untuk Meminimasi Consumption Energy Pada Penjadwalan Flow Shop. *Prosiding SENTRA (Seminar Teknologi dan Rekayasa) 4*, 47-54.

[18] D. M. Utama, D. S. Widodo, W. Wicaksono, and L. R. Ardiansyah. (2019). A New Hybrid Metaheuristics Algorithm for Minimizing Energy Consumption in the Flow Shop Scheduling Problem. *International Journal of Technology 10(2)*, 320-331.

[19] Y. Liu, H. Dong, N. Lohse, and S. Petrovic. (2016). A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance. *International Journal of Production Economics 179*, 259-272.

[20] X. Liu, L. Wang, L. Kong, F. Li, and J. Li. (2019). A Hybrid Genetic Algorithm for Minimizing Energy Consumption in Flow Shops Considering Ultra-low Idle State. *Procedia CIRP 80*, 192-196.

[21] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production 144*, 228-238.

[22] D. Tang, M. Dai, M. A. Salido, and A. Giret. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry 81*, 82-95.

[23] Y. Yun, E. J. Hwang, and Y. H. Kim. (2019). Adaptive genetic algorithm for energy-efficient task scheduling on asymmetric multiprocessor system-on-chip. *Microprocessors and Microsystems 66*, 19-30.

[24] J.-f. Chen, L. Wang, and Z.-p. Peng. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation 50*, 100557.

[25] M. Sharma and R. Garg. (2019). HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers. *Engineering Science and Technology, an International Journal*.

[26] C. Expósito-Izquierdo, F. Angel-Bello, B. Melián-Batista, A. Alvarez, and S. Báez. (2019). A metaheuristic algorithm and simulation to study the effect of learning or tiredness on sequence-dependent setup times in a parallel machine scheduling problem. *Expert Systems with Applications 117*, 62-74.

[27] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software 114*, 163-191.

[28] S. Li, F. Liu, and X. Zhou. (2018). Multi-objective energy-saving scheduling for a permutation flow line. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 232(5)*, 879-888.

[29] H. T. Ibrahim, W. J. Mazher, O. N. Ucan, and O. Bayat. (2017). Feature Selection using Salp Swarm Algorithm for Real Biomedical Datasets. *IJCSNS 17(12)*, 13.

[30] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili. (2018). Asynchronous accelerating multi-leader salp chains for feature selection. *Applied Soft Computing 71*, 964-979.

[31] C. Nayak, S. K. Saha, R. Kar, and D. Mandal. (2018). Optimal SSA-based wideband digital differentiator design for cardiac QRS complex detection application. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, e2524.